MATTHEW D. POWERS (Bar No. 104795)
E-Mail: matthew.powers@weil.com
EDWARD R. REINES (Bar No. 135960)
E-Mail: edward.reines@weil.com
JEFFREY G. HOMRIG (Bar No. 215890)
E-Mail: jeffrey.homrig@weil.com
JILL J. HO (Bar No. 236349)
E-Mail: jill.ho@weil.com
WEIL, GOTSHAL & MANGES LLP
Silicon Valley Office
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

Attorneys for Plaintiff-Counterclaim Defendant
NETAPP, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

NETWORK APPLIANCE, INC.,

Plaintiff-Counterclaim Defendant,

v.

SUN MICROSYSTEMS, INC.,

Defendant-Counterclaim Plaintiff.

Case No. C-07-06053 EDL

DECLARATION OF GREG GANGER, Ph.D. IN SUPPORT OF NETAPP'S RESPONSE TO SUN'S OPENING CLAIM CONSTRUCTION BRIEF

I, Greg Ganger, Ph.D., declare:

1. I have been retained by Plaintiff-Counterclaim Defendant NetApp, Inc. ("NetApp"), in this case. If called upon as a witness, I could competently testify to the truth of each statement herein.

I.

QUALIFICATIONS

2. I am a Professor of Electrical and Computer Engineering at Carnegie Mellon University, which includes teaching, research, and advising responsibilities across a range of topics, including storage and file systems. I also serve as the Director of the Parallel Data Lab (PDL) at Carnegie Mellon, which is a research center focused on storage and file systems. My qualifications to render an expert opinion in the matter are set forth in my Curriculum Vitae, which is attached as Exhibit A. My C.V. also contains a list of publications authored in at least the last 10 years.

II.

STATEMENT OF OPINIONS - U.S. PATENT NO. 5,819,292

A. ORDINARY SKILL IN THE ART

3. One of ordinary skill in the art in 1993, the effective filing date of the '292 patent, would generally have a bachelor's or master's degree in Computer Engineering or Computer Science, or equivalent experience, and several years experience in working in the area of file systems and/or data storage systems.

4. My opinion is based upon my personal knowledge and experience and my consideration of the following factors: (1) the levels of education and experience of persons of skill working in the field; (2) the sophistication of the technology and types of problems encountered; (3) prior art; and (4) the rapidity with which innovations are made in this field.

B. BACKGROUND

5. The '292 patent describes methods for maintaining a file system with various novel features. The file system organizes raw storage capacity into blocks holding (1) "meta-data" that describes the file system, and (2) raw contents of files. At the root of the tree of blocks is a structure called a "root inode." A set of self-consistent blocks on disk that is rooted by the root

inode is referred to as a "consistency point," a simplified example of which is shown in the figure below:

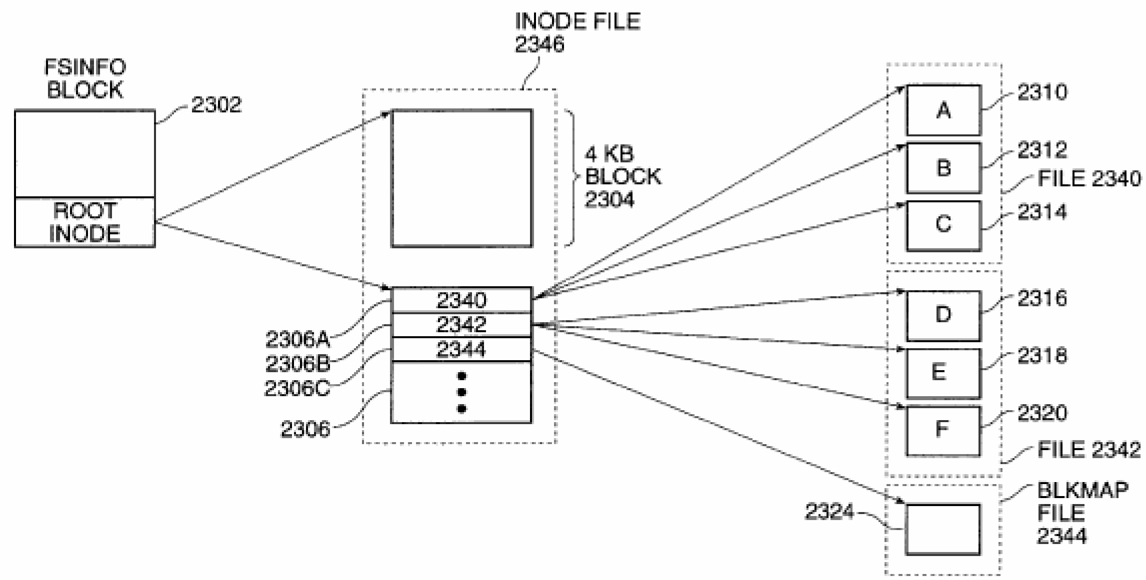

FIG. 17B

'292 patent at Fig. 17B.

6. In the above exemplary figure, the meta-data includes the "inode file" and "blkmap file," while raw data are in the blocks labelled A-F. The inode file includes one "inode" for each file, other than the inode file itself. *Id.* at 9:29-32. Similar to how the root inode roots the entire file system, an inode roots a tree of blocks corresponding to a single file. The blkmap file represents one way of keeping track of which blocks in the file system are in use. *See id.* at 9:50-65.

7. The '292 patent describes a method of moving the file system from one self-consistent state to another self-consistent state as modifications are made by applications, users, and the like. Briefly, when a user modifies data in one of the data blocks (blocks A-F in the figure above), the file system does not overwrite the previous block in storage. Instead, it stores the new data in a memory buffer, allocates a new block in storage for it, and writes it to the new location. The original block (from the previous consistent state) remains unchanged in storage. *Id.* at 12:3-5. The block in the tree that points to this new block must, in turn, also be modified to reflect the new location – this block too is written to a new location. So, the block that points to that block must be modified and written to a new location.. And, so on, for all the blocks in the tree leading

to the modified block, all the way up to (but not including) the root inode. Eventually, all of the modified buffers that were held in memory are flushed to storage along with a new version of the root inode. However, the blocks of the previous state of the file system (*i.e.*, the "consistency point") remain on disk. These blocks are never overwritten when updating the file system to ensure that both the old consistency point and the new consistency point exist on disk until finally transitioning from one to the other (by writing the root inode). *See id.* at 17:56-63. By having a root inode that anchors an entire file consistent state of a file system, and not overwriting modified blocks (except for the root inode, when transitioning to the next consistent state), the entire file system can be preserved in a consistent state at all times. A similar approach can be used to make a series of read-only copies of the entire file system, referred to as "snapshots," by copying the root inode during the transition to a new consistent state and marking which blocks are part of which snapshots. *See id.* at 17:65-23:18.

C. THE DISPUTED TERMS

1. "non-volatile storage means"

8. One of ordinary skill in the art at the time the '292 patent was filed would have understood that the claim term "non-volatile storage means" refers to a storage device that can retain information in the absence of power. Furthermore, one of ordinary skill in the art would have understood that, in the context of the '292 patent, the term "non-volatile storage" refers to any of a set of well-known devices upon which file systems may be maintained, including disks, disk arrays, flash memory drives, and the like.

9. I am informed that patent claims may include limitations that recite a means or step for performing a specified function without the recital of structure. Such limitations are then understood to cover the corresponding structure or material described in the specification and any equivalents thereof. I am informed that, although a claim term using the word "means" is generally presumed to be such a "means-plus-function" limitation, the mere use of the word "means" does not require that limitation to be a "means-plus-function limitation." I understand that a claim element that uses the word "means" but recites no function corresponding to the means is not a means-plus-function limitation. Also, even if the claim element specifies a

function, the claim element is still not a mean-plus-function limitation if it also recites sufficient structure or material for performing that function.

10. One of ordinary skill in the art would understand that the term "non-volatile storage means," as used in the '292 patent, is not a "means-plus-function" limitation, whether starting with a presumption that it is or not. First, the claim term is not linked to a function. In its opening brief, Sun asserts that "non-volatile storage means" corresponds to the functions of (1) storing blocks of a file system (claims 4 and 8), (2) storing first and second "file information structures" (claim 4), (3) storing "read-only copies of a file system" (claim 8), and (4) storing "metadata for successive states of said file system" (claim 8). *See* Sun's Opening Cl. Constr. Br. at 8-9. However, one of ordinary skill in the art, upon reviewing the claims, would understand that the "non-volatile storage means" is not the thing carrying out these functions, several of which correspond to steps of method claims 4 and 8. Rather, one of ordinary skill in the art would understand that the thing carrying out these functions is the component implementing the claimed methods (*e.g.*, software executing on a file server).

11. One of ordinary skill in the art would understand this because the claim language explicitly describes these functions as being performed by something else acting on the passive "non-volatile storage means." For example, Sun's second proposed function is recited in the claims as "*storing in* said non-volatile storage means a second file system information structure." *See* '292 patent at 25:25. One of ordinary skill in the art would understand that this means that the "non-volatile storage means" is a passive component into which something is stored, rather than an active component performing the described step of "storing." Further, claim 4 recites "comprising blocks of data stored in blocks of a non-volatile storage means," in which the blocks are "stored in" something (the non-volatile storage means), but there does not appear to be any function recited at all. *See id.* at 25:12-14. In its second step, Claim 4 recites "writing blocks of data . . . to . . . said non-volatile storage means," again showing that the non-volatile storage means is merely where the blocks of data are being written, not performing the act of writing itself. *See id.* at 25:21-24. Indeed, the "writing to" is a function that a "non-volatile storage means" clearly does not do, since it is the thing that is being *written to*. Thus, one of ordinary

skill in the art would conclude that both the "writing" and "storing" are done by the component implementing the method, which is *not* the "non-volatile storage means." These arguments apply equally to the other functions that Sun says the claims describe the "non-volatile storage means" as doing.

12. Even if the claims recited a function for the "non-volatile storage means," which they do not, one of ordinary skill in the art still would not understand the term to be a means-plus-function limitation, because the term "non-volatile storage" has a well understood meaning in the art. Like others of skill in the art, I understand this term to denote a storage device that can retain information in the absence of power, such as a disk, disk array, flash memory drive, and the like. For example, I have considered the following dictionary definitions, which all confirm my opinion:

- The *IEEE Standard Dictionary of Electrical and Electronics Terms* defines "nonvolatile storage" as "a storage device which can retain information in the absence of power; a type of storage whose contents are not lost when power is lost." Exh. D (*IEEE Standard Dictionary of Electrical and Electronics Terms* 699 (6th ed. 1996)).
- The *IEEE Standard Dictionary of Electrical and Electronics Terms* dictionary defines "nonvolatile memory" to be "a storage system that does not lose data when power is removed from it." *Id.*
- The *Prentice Hall's Illustrated Dictionary of Computing* defines "nonvolatile storage" as "a storage device whose contents are not lost when the power is cut off." Exh. E (*Prentice Hall's Illustrated Dictionary of Computing* 416 (2d ed. 1995)).
- The *Prentice Hall's Illustrated Dictionary of Computing* defines "nonvolatile memory" to be "memory chips (such as read-only memory) which retain their information even after the electrical power has been switched off. This contrasts with volatile chips (such as random access memory) which lose their information when the power is switched off." *Id.*
- The *Microsoft Press Computer Dictionary* defines "nonvolatile memory" to be, "a storage system that does not lose data when power is removed from it. Intended to refer to core, ROM, EPROM, bubble memory, or battery backed CMOS RAM, the term is occasionally used in reference to disk subsystems as well." Exh. F (*Microsoft Press Computer Dictionary* 241 (1991)); *see also* Exh. G (*Microsoft Press Computer Dictionary* 332 (3d ed. 1997)) (providing the same definition).
- The *Webster's New World Dictionary of Computer Terms* defines "nonvolatile memory" to be "the memory specifically designed to hold information, even when the power is switched off. Read only memory (ROM) is non-volatile, as are all secondary storage units such as disk drive." Exh. H (*Webster's New World Dictionary of Computer Terms* 376 (8th ed. 2000)).
- The *McGraw-Hill Dictionary of Scientific and Technical Terms* defines "nonvolatile storage" to be a "computer storage medium that retains

information in the absence of power, such as magnetic tape, drum, or core. Also known as nonvolatile memory." Exh. I (*McGraw-Hill Dictionary of Scientific and Technical Terms* 1439 (6th ed. 2003)).

- The *Webster's New World Dictionary of Computer Terms* explains that "nonvolatile storage" is a "storage medium that retains its data in the absence of power, such as *ROM*." Exh. J (*Webster's New World Dictionary of Computer Terms* 256 (3d ed. 1988)).
- The *Dictionary of Computer Words* defines "nonvolatile memory" to be, in part, "[m]emory whose contents are not lost when the system power is shut off." This dictionary further explains that "[d]isk storage and *ROM* are both nonvolatile memory, as opposed to the volatile memory held in *RAM*." Exh. K (*Dictionary of Computer Words* 170 (1993)).
- The *Ilustrated Dictionary of Computer Words* defines "nonvolatile storage" to be, in part, "[s]torage medium that retains its data in the absence of power, such as *magnetic bubble memory* and *magnetic core storage*." Exh. L (*Ilustrated Dictionary of Computer Words* 203 (3d ed. 1986)).

13. Notably, all of these definitions are consistent in defining "non-volatile storage" or "non-volatile memory" to be storage that can retain data in the absence of power. Furthermore, all explicitly define it with structural language, as a "device", a "storage system", "a medium", or the like. Several even give explicit examples of such devices. For example, the *Illustrated Dictionary of Computer Words* explains that "magnetic bubble memory" or "magnetic core storage" may be "non-volatile storage." Likewise, the *McGraw-Hill Dictionary of Scientific and Technical Terms* gives "magnetic tape, drum, or core" as examples of "non-volatile storage" and explains that it is "[a]lso known as non-volatile memory." Similarly, the *Microsoft Press Computer Dictionary* explains that "non-volatile memory" is "[i]ntended to refer to core, ROM, EPROM, bubble memory, or battery backed CMOS RAM" and that "the term is occasionally used in reference to disk subsystems as well." *See* Exh. F (*Microsoft Press Computer Dictionary* 241 (1991)). Also, the definition of "non-volatile memory" in the *Websters's New World Dictionary of Computer Terms* explains that "[r]ead only memory (ROM) is non-volatile, as are all secondary storage units such as disk drives." Exh. G (*Webster's New World Dictionary of Computer Terms* 376 (8th ed. 2000)).

14. Sun's expert, Dr. Brandt, opined that the term "non-volatile storage means" is too broad to be meaningfully understood by someone of ordinary skill in the art in the context of this patent. I disagree. In fact, it is instructive that Sun and Sun's expert, Dr. Brandt, both exhibit comfort with

the very similar term, "permanent storage," which is often used interchangeably with "non-volatile storage" in the art. Specifically, Sun's opening brief, when describing the background to the '211 patent, notes that "data is written to *permanent storage, such as a disk*." Sun's Opening Cl. Constr. Br. at 19. Dr. Brandt's declaration repeats this language almost word-for-word. *See* Brandt. Decl. ¶ 101. This indicates to me that people of skill in the art (*e.g.*, for instance, Dr. Brandt) and those trying to explain the art (*e.g.*, Sun) understand that, when the '292 patent (which shares the same specification as the '211 patent) refers to "non-volatile storage," it is referring to well-known structural components used for storing blocks of file systems, "such as a disk."

15. In addition, I note that Marshall Kirk McKusick, an expert that Sun retained in this matter, also understands that the term generally refers to a computer disk. Indeed, in a very recent publication describing the development of the BSD file system, Dr. McKusick equates "nonvolatile storage" with "disk," stating that "information in nonvolatile storage (*i.e.*, disk) must always be consistent . . . ." *See* Exh. M (Marshall Kirk McKusick, *A Brief History of the BSD Fast File System*, ;login: The Usenix Magazine, June 2007) at 12. Likewise, in a publication that I co-authored with Dr. McKusick, we described "non-volatile storage" in an identical way. *See* Exh. N (Gregory R. Ganger et al., *Soft Updates: A Solution to the Metadata Update Problem in File Systems*, ACM Transactions on Computer Systems 127 (2000)) at 128 ("[T]he information in nonvolatile storage (*i.e.*, disk) must always be consistent . . . .").

16. Along these lines, Sun's own patent portfolio reveals many patents – too many to list exhaustively – in which Sun apparently had no trouble understanding the meaning of "non-volatile storage" and equating it with a structure, usually a disk:

- "The IN-Mod must be updated (for a given inode) before it is committed to *non-volatile storage (i.e. disk or NVRAM)* . . . ." Exh. O (U.S. Patent No. 7,089,293) at 47:14-15.
- "*[N]on-volatile storage* . . . may include a floppy disk drive, a RAM card, a hard drive, CD-ROM drive, or other magnetic, re-writable optical, or other mass storage devices" Exh. P (U.S. Patent No. 5.954,826) at 6:66-7:2.
- "In the server, there is a large capacity *non-volatile storage device, such as a hard disk drive* . . . ." Exh. Q (U.S. Patent No. 5,721,824) at 1:22-23.
- "[A] data storage system includes a computer coupled to a *non-volatile storage, such as a disk drive* . . . ." Exh. R (U.S. Patent No. 6,629,198) at 2:2-

4.

- "[T]he processor's internal context is saved in RAM or non-volatile memory (*e.g.*, disk storage) . . . ." Exh. S (U.S. Patent No. 5,878,264) at 10:28-29.
- "Storage 164, such as a computer disk drive or other nonvolatile storage may provide storage of data or program instructions." Exh. T (U.S. Patent No. 5,929,792) at 4:34-36.

17. Likewise, both Sun's Brief and Dr. Brandt's declaration identify a number of items that may reasonably correspond to "non-volatile storage." *See* Sun's Opening Cl. Constr. Br. at 7; Brandt Decl., ¶ 65. In an attempt to suggest that the term "non-volatile storage" is too broad to be reasonably understood by one of ordinary skill in the art, Sun and Dr. Brandt constructed as lengthy a list of items as possible. I do agree that the term "non-volatile storage" should be understood broadly to include, in addition to disks, other components on which file systems may be maintained, such as flash memory drives, disk arrays, battery backed RAM devices, and the like. But, I believe Sun and Dr. Brandt go too far in suggesting that relevant "non-volatile storage" includes things like "paper" and "film" simply because they retain data in the absence of power. One of ordinary skill in the art would understand that, in the context of the patent, which focuses on maintaining consistent states and read-only copies of active file systems, the term "non-volatile storage" does not refer to things like "paper" and "film," which are not used for maintaining active (computer) file systems. Indeed, in describing the many "storing" functions that Sun incorrectly ascribes to the "non-volatile storage means," the preferred embodiment descriptions in the '292 patent's specification repeatedly explain that the thing in which the various file system structures are stored is a "disk." *See, e.g.*,'292 patent at 9:48-49 (explaining that the "inode file," which is metadata, is "written *to disk*"); *id.* at 12:2-3 ("WAFL always writes new data to unallocated blocks *on disk*"); *id.* at 11:66-12:1 ("set [or tree] of self-consistent blocks *on disk*"); *id.* at 17:65-18:7 ("read-only copy of an entire file system" stored on disk); *id.* at 12:18-19 ("root inode . . . is written *to disk*"). One of ordinary skill in the art would have little trouble recognizing the clear parallels between "non-volatile storage means" in the claims and "disk" (by far, the most common example of non-volatile storage in the field) in the specification.

18. Even if this term is understood to be a means-plus-function limitation, one of ordinary skill in the art would understand that Sun's proposed construction for "non-volatile storage

means," which asks for unnecessary and irrelevant aspects like a "4KB block size" with "no fragments," is too narrow. In my opinion, Sun's proposed construction suffers from two major problems. First, Sun's construction asks that the underlying characteristics of the "non-volatile storage" be understood to include characteristics of the overlying file system that organizes and uses the raw capacity of the "non-volatile storage means." Doing so confuses jars and apple sauce (although the latter can be put in the former, it in no way defines it) and would make little sense to one of ordinary skill in the art. Second, even if it were reasonable to apply file system characteristics to the characteristics of the "non-volatile storage means," the characteristics that Sun attributes the "non-volatile storage means" are not required to carry out any of the functions Sun ascribes to it.

19. Sun's proposed construction suggests that the term "non-volatile storage" be understood to require (1) 4KB blocks, (2) blocks with no fragments, and (3) disk storage blocks with the same size as the data blocks of the file system. After reviewing the specification, however, one of ordinary skill in the art would understand that these characteristics do not pertain to the "non-volatile storage" device. Rather, Sun is taking limitations from the specification pertaining to how the *file system* organizes the raw storage capacity of a "non-volatile storage" device (a disk, in the preferred embodiment examples) and applying them to the underlying device. This is evident from the fact that Sun draws its proposed structure from a section of the specification describing the "*File*-system Layout" *See* '292 patent at 5:47-60; *see also* Sun's Opening Cl. Constr. Br. at 9. One of ordinary skill in the art would understand, from this heading alone, that this section focuses not on the "non-volatile storage device," but rather on how the overlying file system organizes the raw capacity.

20. This is confirmed if one reviews the complete sentence from which Sun draws "4KB blocks that have no fragments," located at the front of the description of the "File-System Layout," which is as follows:

> "The present invention uses a Write Anywhere File-system Layout (WAFL). This disk format system is block based (*i.e.*, 4 KB blocks that have no fragments), *uses inodes to describe its files, and includes directories that are simply specially formatted files*.

’292 patent at 5:49-52 (Emphasis added). The italicized text (which deals with “inodes,” “files,” and “directories”) does not contain any language that one of ordinary skill in the art would understand to describe characteristics of the underlying “non-volatile storage” device. Instead, as the section header and preceding sentence pronounce, this entire passage (including the phrase, “4KB blocks that have no fragments”) refers to characteristics of the preferred embodiment file system (WAFL) and not to its underlying “non-volatile storage means.” Indeed, the concept of block “fragments” is a well-understood file system concept, not a characteristic of a disk or other “non-volatile storage means.” A file system that supports fragments uses them to allow the contents of multiple small files to be packed into a single block – the block is divided into fragments that can each be allocated separately.

21. Even if this term were a means-plus-function limitation, and even if it were not unreasonable to apply file system characteristics to an underlying “nonvolatile storage means,” one of ordinary skill in the art would still understand that Sun’s proposed construction is overly restrictive.

22. First, if the term “non-volatile storage means” were functional, one of ordinary skill in the art would conclude that the term simply refers to a “means” for providing “non-volatile storage.” “Non-volatile storage” clearly means a storage device (a container) that can retain information in the absence of power. Numerous dictionary definitions confirm this, and Dr. Brandt had no difficulty identifying examples of non-volatile storage devices. Nor did Sun in its Opening Cl. Constr. Br. or in many of its issued patents. *See supra* Part II.C.1.a. Thus, to the extent this term requires a corresponding function, one of ordinary skill in the art would understand that that function is “retaining information so that the information is not lost in the absence of power.” Requiring the function to also include “blocks of data in a file system” is unnecessary, because other claim terms describe what is stored in the non-volatile storage means, such as the “blocks of data.”

23. To the extent this function requires a corresponding structure, one of ordinary skill in the art would immediately recognize the “disk” discussed frequently in the specification as where blocks are stored in the preferred embodiment. *See, e.g.*, ’292 patent at 12:2-3 (“WAFL always

writes new data to unallocated blocks *on disk*"); *id.* at 11:66-67 ("set [or tree]of self-consistent blocks *on disk*"). One of ordinary skill in the art would also understand from the specification that the particular attributes of the "non-volatile storage," such as disks, should be understood broadly:

> In the following description, numerous specific details, such as *number and nature of disks*, disk block sizes, etc., are described in detail in order to provide a more thorough description of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details.

Id. at 5:37-42 (Emphasis added). Based on this, one of ordinary skill in the art would understand that a "non-volatile storage means" should not be limited to a specific "nature of disks" that requires such details as a precise 4KB block size. Certainly, it should not be understood to require (1) 4KB blocks, (2) blocks with no fragments, and/or (3) disk storage blocks with the same size as the file system data blocks, which, as explained above, are characteristics of the overlying file system rather than the "non-volatile storage means." Indeed, these details are not required to carry out Sun's overly restrictive definition of the claimed function, which is "storing blocks of data of a file system so that the data is not lost in the absence of power." For this, all that is required is a computer disk, or any other non-volatile storage device, such as those referred to by Dr. Brandt, in the various Sun patent cited above, and in the many dictionary definitions that define "non-volatile storage." *See supra* Part II.C.1.a.

2. "meta-data for successive states of said file system"

24. One of ordinary skill in the art in 1993, the effective filing date of the '292 patent, would have understood that the claim term "meta-data for successive states of said file system" refers to information that describes successive states of a file system. Both the claims and the specification make this clear.

25. Sun and Dr. Brandt argue that this claim term means "a block map file for recording snapshots of the file system." *See* Sun's Opening Claim Construction Brief at 10; Brandt Decl. ¶ 68. I disagree.

26. Claim 8 of the '292 patent recites:

> 8. A method for creating a plurality of read-only copies of a file system stored in blocks of a non-volatile storage means, said file system comprising meta-data identifying blocks of said non-volatile storage means used by said file system, comprising the steps of:
>
> storing **meta-data for successive states of said file system** in said non-volatile storage means;
>
> making a copy of said meta-data at each of a plurality of said states of said file system;
>
> for each of said copies of said meta-data at a respective state of said file system, marking said blocks of said non-volatile storage means identified in said meta-data as comprising a respective read-only copy of said file system.
>
> ’292 patent at 26:1-15 (Emphasis added).

27. One of ordinary skill in the art would understand that this method for creating snapshots has three steps: (1) storing meta-data for consistency points; (2) copying the meta-data at each consistency point for which snapshots are created; and (3) marking which blocks correspond to which snapshot(s).

28. One of ordinary skill in the art would understand that “successive states of said file system” refers to consistency points, not snapshots. Indeed, it is my understanding that the parties have agreed that “consistent state” and “state of a file system” are synonymous.

29. Furthermore, one of ordinary skill in the art would have understood that the word “meta-data” is not limited to a block map file. On the contrary, “meta-data” is commonly understood to include many structures that contain information about the corresponding data.

30. The specification of the ’292 patent supports the broad meaning of the term “meta-data.” It lists several examples of meta-data, including an inode file, a root inode, a block map file, an inode map file, inode tables, directories, bitmaps, and indirect blocks. *Id.* at 9:18-10:56; 1:38-39.

31. I am informed and I understand that dependent claims are generally narrower in scope than the independent claims from which they depend.

32. The dependent claims in the ’292 patent, particularly claims 11, 12, 13, 18 and 19, confirm the fact that “meta-data” has a broad meaning. Those claims require that “meta-data for successive states of said file system” be at least broad enough to include pointers to a hierarchical tree of blocks, structures representing files of the file system (*e.g.*, inodes), and a root structure

referencing structures representing files of said file system (*e.g.*, a root inode).

33. I am informed that an additional limitation added by a dependent claim generally should not be read into the independent claim from which it depends, especially when the limitation in dispute is the only meaningful difference between the two.

34. Given this understanding, dependent claims 9 and 10 further confirm my opinion that "meta-data for successive states of said file system" is not limited to a block map file.

35. I understand that Sun argues that dependent claims 9 and 10 further define the type of block map file required by claim 8. *See* Sun's Opening Cl. Constr. Br. at 10-11. I disagree.

36. A person of ordinary skill in the art would not understand claim 8 to require a "means for recording multiple usage bits per block," because that limitation is added by claim 9. Claim 8 would certainly not be understood to require the more limited example "means for recording multiple usage bits per block" that is a "block map comprising multiple bit entries for each block," because that is the limitation added to claim 9 by claim 10. Indeed, there are other ways besides a block map to keep track of which blocks correspond to which snapshot(s). For example, bits could be maintained with each pointer (in inodes and indirect blocks) to identify which snapshots, if any, the pointed to block is part of. Or, the file system could simply keep a list of which blocks are used in each snapshot.

37. More importantly, a person of ordinary skill in the art would understand that claims 9 and 10 add limitations to the "marking" step of claim 8, and do not relate to the "storing meta-data" step at all.

38. Therefore, there is no reason to think that the explicit reference to a block map in claim 10 is relevant to the meaning of "meta-data for successive states of said file system."

39. A person of ordinary skill in the art reading the '292 patent would recognize that the WAFL system is a preferred embodiment.

40. Dr. Brandt's declaration points to various portions of the specification that describe the block map file used in the preferred embodiment. *See* Brandt Decl., ¶¶ 71-75. I disagree that these passages support Sun's interpretation of "meta-data for successive states of said file system."

41. In the preferred embodiment, the creation of a snapshot involves copying a root inode, not copying a block map file. *See, e.g.*, '292 patent at 18:13-16; 18:65-19:6. There is also a block map file that keeps track of whether each block is part of the active file system and/or one or more snapshots. But, although the block map file is updated, it is not copied when a new snapshot is created.

3. "file system information structure"

42. One of ordinary skill in the art in 1993, the effective filing date of the '292 patent, would have understood that the claim term "file system information structure" refers to a data structure containing information about the layout of a file system. Both the claims and the specification make this clear.

43. Sun and Dr. Brandt argue that this claim term means "data structure that contains the root inode of a file system in a fixed location on a disk" *See* Sun's Opening Cl. Constr. Br. at 14; Brandt Decl. ¶ 77. I disagree.

44. Claim 4 of the '292 patent recites:

> 4. A method for maintaining a file system comprising blocks of data stored in blocks of a non-volatile storage means at successive consistency points comprising the steps of:
>
> storing a first **file system information structure** for a first consistency point in said non-volatile storage means, said first **file system information structure** comprising data describing a layout of said file system at said first consistency point of said file system;
>
> writing blocks of data of said file system that have been modified from said first consistency point as of the commencement of a second consistency point to free blocks of said non-volatile storage means;
>
> storing in said non-volatile storage means a second **file system information structure** for said second consistency point, said second **file system information structure** comprising data describing a layout [of] said file system at said second consistency point of said file system.
>
> '292 patent at 25:12-29. at 25:12-29 (Emphasis added).

45. A "file system information structure" is a data structure that contains information about a file system. One of ordinary skill in the art reading the '292 patent would understand that, while other types of information about the file system (*e.g.*, the number of blocks in the file system, the creation time, and a checksum) might also be included, information about the layout of the file

system at a given consistency point (as recited in claim 4) would certainly be included in the file system information structure.

46. As mentioned above, I am informed that an additional limitation added by a dependent claim should not normally be read into the independent claim from which it depends, especially when the limitation in dispute is the only meaningful difference between the two.

47. Given that understanding, claims 5 and 6 support my opinion that a file system information structure need not be stored at a fixed location. In particular, it would not be necessary or appropriate for a dependent claim to recite a fixed location for file system information structures, if a fixed location were part of the definition of that term.

48. I also disagree with Sun's argument, and Dr. Brandt's opinion, that a file system information structure necessarily contains a root inode. *See* Sun's Opening Cl. Constr. Br. at 15-16; Brandt Decl. ¶¶ 78, 81. A person of ordinary skill in the art would understand that any data structure representative of the entire file system, not just a root inode, could be stored in the file system information structure. In fact, the specification confirms that the invention does not require the file system information structure to include a root inode:

> "Snapshots are created by duplicating the root data structure of the file system. In the preferred embodiment, the root data structure is the root inode. However, **any data structure representative of an entire file system could be used**."
>
> '292 patent at 18:13-16 (Emphasis added) .

49. Sun and Dr. Brandt argue that "file system information structure" and "file system information block" are used interchangeably. *See* Sun's Opening Cl. Constr. Br. at 14-15; Brandt Decl. ¶ 79. I disagree.

50. The phrase "file system information block" is used in two ways in the '292 patent specification. Sometimes it is used to refer to a data structure, and sometimes it is used to refer to a particular location where that data structure is stored in the preferred embodiment. Where "fsinfo block" is used to refer to the data structure itself, it is synonymous with "fsinfo structure."

51. One of ordinary skill in the art would understand which meaning of "file system information block" was intended based on the context of different passages in the specification.

For example, "file system information (fsinfo) block" is used to refer to a location at 9:33-35 ("a fixed location on disk referred to as the file system information (fsinfo) block described below"), but it refers to the data structure itself at 13:66-67 ("The fsinfo block is written twice. It is first written to one location and then to a second location.").

52. Unlike "file system information block," "file system information structure" has only one meaning in the specification: that of the data structure itself (not its location).

III.

STATEMENT OF OPINIONS – U.S. PATENT NO. 6,892,211

53. The '211 patent is a continuation of the '292 patent. Likewise, my opinions expressed above regarding the hypothetical person of ordinary skill in the art and the general subject matter of the '211 patent are the same as for the '292 patent. *See* ¶¶ 4-7.

A. THE DISPUTED TERMS

1. "pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system"

54. One of ordinary skill in the art in 1993, the effective filing date of the '211 patent, would have understood that this claim term refers to pointing to a group of things (buffers in memory and blocks on storage) using some combination of direct and indirect references. More precisely, it means pointing directly to blocks and/or buffers, and/or indirectly to blocks and/or buffers. Both the claims and the specification make this clear.

55. Sun and Dr. Brandt argue that this claim term means "pointing directly and indirectly to buffers in said memory and pointing directly and indirectly to a second set of blocks on said storage system" *See* Brandt. Decl ¶ 102. I disagree.

56. As an example, Claim 1 of the '211 patent recites:

> A method of maintaining a file system stored in a memory and on a storage system that includes one or more hard disks, said method comprising steps of:
>
> maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of nblocks on said storage system that store a first consistent state of said file system; and
>
> maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set

> of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.
>
> *See* '211 patent at 23:61-24:11.

57. I do not believe any construction is necessary for this term, as its meaning seems apparent from plain interpretation of the words.

58. To the extent that a construction is deemed necessary, it should be consistent with the relatively straightforward concept being conveyed by the claim term. Specifically, the claims that use it describe two root inodes each pointing to a group of data containers holding the contents of a consistent state of a file system. The first, an on-disk root inode, points to a consistent state for which all contents are on disk (in blocks). The second, an incore root inode, points to a consistent state for which some of the contents may be in memory (in buffers) and the remainder are on disk (in blocks). Each root inode uses some combination of direct and indirect pointers in pointing to the group of containers that represents its respective consistent state. The disputed claim term is about the second (incore) root inode, which is required to point to some combination of buffers and blocks using some combination of direct and indirect pointers.

59. From Sun's opening brief, it is clear that they do not assert that the claim term requires that the incore root inode be pointing both directly and indirectly to any particular container (block or buffer). For example, on page 23, they discuss several examples in which there are direct pointers to some containers and indirect pointers to others. Indeed, the specification text (*e.g.*, 6:23-52 and 8:26-56) and many of the figures (*e.g.*, Figures 4B, 4C, 9B, 9C) preclude such an interpretation, as it would cause the claims to not cover the preferred embodiment. So, it is agreed that no single container from the group must be pointed to directly and indirectly.

60. Sun's proposed construction requires partitioning the group of containers storing the second consistent state into two subgroups (the blocks and the buffers) and then requiring that there be both direct and indirect pointers to each subgroup. This interpretation is inconsistent with the clear indication that it is a singular group, instances of the preferred embodiments of the

specification, and Sun's existing confirmation that a single container need not be pointed to directly and indirectly.

61. Sun's interpretation is inconsistent with the claim language, which makes clear that the "blocks and buffers" being pointed to are one group and not two. The relevant portion of claim 9 is as follows:

> [W]herein said memory also stores information including instructions executable by said processor to maintain said file system, the instructions including the steps of
>
> (a) maintaining an on-disk root inode on said storage system, said *on-disk root inode pointing directly and indirectly to a* ***first set of blocks*** *on said storage system* ***that store a first consistent state of said file system***, and
>
> (b) maintaining an incore root inode in said memory, said *incore root inode pointing directly and indirectly to* ***buffers in said memory and a second set of blocks*** *on said storage system,* ***said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system***, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.
>
> *See* '211 patent at 24:45-62 (emphasis added).

As the emphasized portions of the above quotation show, there is a parallel structure in this claim. The first set of blocks (described at "[a]") is the group that represents the first consistent state of the file system. The second set of blocks, together with buffers (described at "[b]"), is the group that represents the second consistent state of the file system. The on-disk root inode and the incore root inode anchor the first and second consistent states of the file system, respectively. Each of these groups is meant to be treated as a whole – they each represent the group of containers storing a consistent state of the file system. There is no indication that the second group (*i.e.*, the blocks and buffers to which the incore root inode points) should be partitioned into subgroups that are treated separately.

62. Indeed, Claim 9 recites, "said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system". The first appearance of the word "and" is clearly being used to establish a single combined group, because the claim describes the "buffers and second set of blocks" together as the group that collectively represents the second

consistent state of the file system. Because this subsequent use of the same phrase makes it clear that the "buffers and second set of blocks" are meant to be treated as a single group, they should not be separated for the purpose of applying each adverb ("directly and indirectly") separately to each.

63. Sun's interpretation is also inconsistent with the specification. Specifically, the preferred embodiments in the specification represent systems in which the incore root inode sometimes does and sometimes does not have both direct and indirect pointers to each type of container in the group (*i.e.*, blocks and buffers). There are circumstances under which a preferred embodiment will have an incore root inode that points to all of the group, but does not point directly to any block or does not point indirectly to any buffer. Clearly, therefore, the specification discloses examples that do not require both forms of pointer for both types of container in the group. Such a limitation should not be invented and inserted into the construction of this term.

64. The simplest example is the case of "not pointing directly to any blocks". An incore root inode described in the specification (*e.g.*, illustrated in Fig. 8 combined with Fig. 3 and discussed in the corresponding text) includes 16 block pointers and 16 buffer pointers. The 16 block pointers are part of the copy of the on-disk root inode that is part of the incore root inode. If all 16 blocks to which the incore root inode points directly are modified, then the 16 buffer pointers will point directly to contents of the second consistent state,[1] but none of the 16 block pointers will do so. The block pointers point to blocks that comprise the inode file as it was in the first consistent state, and they only get updated in the incore root inode during the process of writing the second consistent state to storage. If any of those blocks are modified in forming the second consistent state, there will be buffers that contain the new contents, pointed to directly by ones of the 16 buffer pointers in the incore root inode. As each such change is made, the corresponding block pointer will no longer point to a block that is part of the second consistent state, as it still points to the old block on storage. Thus, if all (up to 16) blocks pointed to directly by the on-disk

[1] In this context, buffers are used to hold contents of blocks that have been modified and no longer match the corresponding blocks (on storage) of the first consistent state.

root inode of the first consistent state change, the incore root inode for the second consistent state will not point directly to any blocks in the second consistent state (until the buffers are written to storage and the incore root inode's block pointers are updated, at which point there may no longer be pointers to buffers).

65. A second example scenario would involve the incore root inode "not pointing indirectly to any buffers." Consider a file system that has only 16 total blocks and 32 total inodes. In the preferred embodiment example of 128-byte inodes and 4KB file system blocks, the 32 inodes would fit within one 4KB inode file block. *See* '211 patent at 9:38-39. In the preferred embodiment example of 32-bit blkmap entries, *id.* at 9:51-53, and ≤ 64-byte files being stored directly in the inode, *id.* at 6:12-14, the blkmap would be stored fully in its inode. Likewise, the preferred embodiment inomap file contents (8 bits per inode file block, *id* at 10:22-24) would fit within its inode. If the only changes between the first consistent state and the second consistent state were to inodes (*e.g.*, to change file permissions or the contents of ≤ 64-byte regular files), then the only buffer that would necessarily be in memory and pointed to (directly or indirectly) by the incore root inode would be that of the one and only inode file block. No other buffers need be part of the group to which the incore root inode points directly or indirectly, meaning that there need not be any indirect pointers to buffers.

66. As a final point, recall that Sun's opening brief recognizes that "pointing directly and indirectly to a group of things" does not require pointing directly and indirectly to any particular item of that group. *See* Sun's Opening Cl. Constr. Br. at 21. Arguing that it does require pointing directly and indirectly to subgroups of the group is only one step removed and is inconsistent with plain reading of the language. It seems clear that, if it is not necessary that one point both directly and indirectly to any member of the group, a plain reading of the language leaves only the interpretation that any combination of direct and indirect pointers that covers the entire group is sufficient.

67. The specification makes clear that there are two types of pointers (direct and indirect) involved in pointing to a set of blocks or buffers and that each is used as appropriate. '211 patent, 5:63-8:57.

68. The claim language, the specification, and plain reading of the claim term are all consistent with the straightforward interpretation: pointing directly to blocks and/or buffers, and/or indirectly to blocks and/or buffers.

2. "root inode"

69. One of ordinary skill in the art at the time the '211 patent was filed in 2004 would have understood that the claim term "root inode" refers to an inode that points directly and/or indirectly to all the blocks in a consistent state of a file system. Both the claims and the specification make this clear.

70. Sun and Dr. Brandt argue that this claim term means "the index node data structure stored in a fixed location that roots a set of self-consistent blocks on the storage system that comprise the file system" *See* Sun's Opening Cl. Constr. Br. at 25; Brandt Decl. ¶ 124. I disagree that the root structure must be stored at a fixed location.

71. My opinions above with respect to the claim term "file system information structure" apply equally here and are incorporated by reference. As described in the preferred embodiment, the root inode can be in the file system information structure, and so the root inode should not be limited to being "in a fixed location" for the same reasons that the file system information structure is not.

72. The claims of the '211 patent recite both "on-disk root inodes" and "incore root inode." One of ordinary skill in the art would understand that "root inode" must be broad enough to accommodate both types of root inodes.

73. The passages cited by Sun describe a root inode stored on disk. A person of skill in the art would not consider those passages relevant to the meaning of an "incore root inode."

74. It is my understanding that the parties agree that "incore" means "in memory."

75. An "incore root inode," like other information temporarily stored in memory, is often kept in whatever available location in memory is allocated by the memory management system.

76. Commonly, memory is allocated and freed dynamically according to need by the software running on a system. Usually, incore inodes are handled by the operating system, and space for them will be allotted in one of two ways: (1) either a routine to allocate memory will be called,

each time space is needed for one, or (2) space for a predetermined number will be set aside during initialization. Even in the latter case, individual locations within the initialization-time space will be assigned to specific inodes on an as-needed basis. So, while a given incore inode (*e.g.*, the incore root inode) may often occupy the same location in memory, there is no fixed location in memory specifically assigned to it.

77. Nothing in the specification suggests that an "incore root inode" is assigned a location in memory differently from other information temporarily stored in memory, much less in a fixed location in memory.

78. Sun and Dr. Brandt argue that the root inode for the active file system must be at a fixed location so that it can be located during initial access to the file system (*e.g.*, after a reboot of the system accessing it). I disagree.

79. One of ordinary skill in the art would understand that there are other mechanisms for ensuring that the root inode for the active file system can be located. For example, rather than storing the root inode itself in a fixed location, the file system could instead store a pointer to the root inode in a fixed location. Or, the file system could have a set of predetermined locations that might hold the root inode – to find the root inode, it would then read all locations in the set to determine which actually held the root inode.

80. Indeed, at the moment a new snapshot is created, at least one copy of the root inode for the current consistency point is not stored in a fixed location.

81. One of ordinary skill in the art reading the '211 patent would understand that root inodes for previous consistency points saved as snapshots are not stored in a fixed location. Indeed, the specification clearly describes that snapshot inodes can be stored anywhere. *See* '211 patent at 18:3-10, 18:58-67, 9:19-29.

IV.

STATEMENT OF OPINIONS – U.S. PATENT NO. 7,200,715

A. ORDINARY SKILL IN THE ART

82. My opinion expressed above regarding the hypothetical person of ordinary skill in the art are the same as for the ’292 patent. *See* ¶ 4.

B. BACKGROUND

83. The ’715 patent is generally focused on improving the read and write efficiency of a RAID array in combination with a file system by increasing the information exchange between the two and exploiting the increased information to enable implicit cooperation between them. RAID, which stands for “Redundant Array of Inexpensive Disks”, involves coupling a number of disks by storing data redundantly across them in an arrangement that allows the contents of one or more failed disks to be recreated from the data on the other disks.[2] RAID technology has enjoyed huge market success in environments that require high-reliability storage.

84. In most RAID configurations, data is “striped” across multiple disks, with one same-sized unit of data on each of the disks. Additional disks are used to store redundancy information, such as parity, computed across the striped data. The corresponding per-disk units of data and redundancy information are often referred to collectively as a “stripe,” and the individual units as “stripe units.”

85. If a disk fails, its contents are lost. But, each lost stripe unit can be reconstructed from a mathematical function (*e.g.*, bitwise addition modulo two) applied to the other stripe units comprising the corresponding stripe. A common RAID configuration of N disks includes N-1 stripe units of data per stripe and one stripe unit of parity to protect data from a single disk failure. Most of my discussion below will focus on this arrangement, though the concepts apply more broadly to the many alternate configurations. For example, each disk may play the same role (*e.g.*, always storing the 3rd data stripe unit or the one parity unit) for every stripe or the roles may be assigned on a stripe-by-stripe basis (*e.g.*, to rotate the parity among the disks or deal with

---

[2] Although RAID was originally conceived for disks, it is well-understood that the same concepts apply to any storage device. For convenience, I may use “disk” and “storage device” interchangeably.

heterogeneous disks). Other RAID configurations can protect against multiple disk failures, either by using multiple disks with redundancy information (*i.e.*, computing multiple equations over the data in the same stripe) or by having each stripe unit of data be part of multiple stripes (*e.g.*, by computing parity as described above plus parity across a different set of stripe units).

86. Although RAID technology is very popular, it is generally hidden behind disk-like interfaces, making it difficult for file systems built atop RAID systems to achieve maximal performance. Each (file system and RAID system) usually focuses its optimization effort on different things, and there is a conflict between those foci. File systems are generally optimized for reading and writing large amounts of related data contiguously, which is what maximizes performance when the underlying storage is a single disk. RAID systems, on the other hand, tend to perform best when writing a full stripe all at once by avoiding time spent reading data from disks in order to recalculate parity. If an entire stripe is written at once, all the blocks in the stripe are in memory, and parity can be calculated without needing to read anything from any of the remaining disks. Using a stripe unit size that accommodates full stripe writes, however, tends to break up the file system's large writes of related data into small pieces spread over a number of disks. Subsequent reads of that related data may then involve many small disk accesses, instead of the single larger access that the file system intended in doing its optimization. Thus, without knowing it, the file system and the RAID system may be working at cross purposes.

87. Ideally, a file system organizing storage provided by a RAID system would be able to simultaneously accomplish both disk I/O optimization and parity recalculation optimization. The inventors of the '715 patent disclose a way of doing that. Specifically, the '715 patent describes a data structure, called an "association," that allows a file system to aggregate multiple write requests. The file system can use information regarding the "topology" of a RAID system (*e.g.*, the number of data disks and the size of the stripe unit) to assign related data to contiguous locations on a *single* disk, while simultaneously assigning other data to the other stripe units of each stripe. The association of multiple requests is sent to the RAID system as a single write request, so that the RAID system can exploit knowledge of the full set of updates in its internal optimization of parity updates. For example, if three full stripes are written, the RAID system can

avoid all reads (as parity can be computed from just the new data) and send a single disk write (to write all three contiguous stripe units) to each disk. Although the file system may sometimes not be able to combine writes to multiple stripes, it can now try to do so with knowledge of the benefits and consequences. Because the file system can map related data to the same disk, while explicitly trying to perform full stripes as well, this idea enables RAID and file system optimization at the same time.

88. In the ’715 patent, the term “storage block” is used to refer to each stripe unit. *See* ’715 patent at 1:37-39. In describing preferred embodiments, the ’715 patent generally discusses the storage blocks as corresponding to the blocks being allocated by the file system (via the block map) to store file system data and metadata. Thus, most preferred embodiment examples use a stripe unit (“storage block”) size that is the same as the file system block size. Most RAID configurations do not use stripe unit sizes that are the same size as the block sizes of the file systems using them, and the correspondence in the preferred embodiment is in no way essential to the ideas introduced in the ’715 patent.

C. THE DISPUTED TERMS

1. “associating the data blocks with one or more storage blocks across the plurality of stripes as an association,” (Claims 21 and 52), and “the association to associate the data blocks with one or more storage blocks across the plurality of stripes” (Claim 39)

89. One of ordinary skill in the art at the time the ’715 patent was filed in 2002 would have understood that these claim terms refer to [creating] a data structure that can relate data blocks to locations on more than one stripe. Both the claims and the specification make this clear.

90. Sun and Dr. Brandt argue that this claim term is indefinite and cannot be construed reasonably but that, if it can be construed, it means “associating each data block with a respective one of the storage blocks across the plurality of stripes as an association.” I disagree on both points.

91. As an example, Claim 21 of the ’715 patent recites:

> 21. A method for controlling storage of data, comprising: receiving one or more write requests associated with data blocks; receiving topological information associated with storage blocks configured in a plurality of parallel stripes of a storage system; **associating the data blocks with one**

> **or more storage blocks across the plurality of stripes as an association**; and writing the data blocks, in response to the association, to the one or more storage devices in a single write request.
>
> *See* '715 patent at 21:41-52.

92. I do not believe any construction is necessary for this term, as its meaning seems apparent from plain interpretation of the words.

93. To the extent that a construction is deemed necessary, it should be consistent with the relatively straightforward concept being conveyed by the claim term. Specifically, there is a data structure (called an "association") that is being used to map one or more data blocks to one or more storage blocks (*i.e.*, stripe units) across the plurality of stripes configured on the storage devices. That is, the data structure identifies the locations where data blocks should be stored, within the space available in multiple stripes.

94. Sun's opening brief argues that this claim term is indefinite because it is impossible for one storage block to be "across a plurality of stripes." *See* Sun's Opening Cl. Constr. Br. at 33. Recall that the '715 patent defines "storage block" to be its term for "stripe unit," which is a part of a stripe by definition. Yet, that definition does not prevent one of ordinary skill in the art from understanding the meaning of the claim terms, because these claim terms do not require that a single storage block be spread across multiple stripes. One of ordinary skill in the art would understand, as a plain reading reveals, that data blocks are being mapped to one or more available locations: the "one or more storage blocks across the plurality of stripes." That is, the group of storage blocks exists across a plurality of stripes, and one or more of them are used by any given "association." Thus, it is possible, such as in cases where only one storage block is in the "association," that only one stripe would be referenced by that "association." Because the claims require an "association" capable of mapping data blocks to storage blocks "across the plurality of stripes," one of ordinary skill in the art would understand that the term "one or more" covers both the degenerate (abnormal) case of a single storage block and the expected case of several storage blocks spread across several stripes. So, it will work in the uncommon case, but it enables writes to multiple stripes.

95. Indeed, the specification frequently describes multi-stripe writes and distinguishes itself

from the prior art on the basis of the prior art not performing writes to multiple stripes. For example, the specification explains that "[p]rior art systems, in contrast, typically send single stripe write transactions to a RAID…" *See* '715 patent at 9:28-29. As another example, the specification explains that "The method includes writing data to a group of storage blocks that include predetermined storage blocks across a plurality of stripes…" *Id.* at 3:14-16. The specification describes another embodiment where, "The storage device manager writes data to a group of storage blocks, which include predetermined storage blocks across a plurality of stripes…" *Id.* at 3:29-31. The specification further describes that, "In the RAID context, some embodiments realize these advantages in part by batching write requests, allocating space in a plurality of stripes to store data of the batched requests, and writing the data to full chains of stripes." *Id.* at 3:66-4:2. The specification further describes the process of assembling the association to include multiple stripes, stating, "The file system layer uses the topology information in a generating block layout information that associates data blocks of the buffered write requests with free storage blocks in the selected stripes, which it then allocates for a write transaction." *Id.* at 5:54-58. The specification also discusses having a file system that will attempt to fill an association and use as much free space on as many stripes as possible. For example, one embodiment includes, "substantially using all available storage blocks in a collection of stripes." *Id.* at 3:7-8. *See also id.* at FIG. 8, which provides a detailed view of an example association that includes data to be written to multiple stripes in a single write request.

96. Thus, the '715 patent is significantly about optimizing *multiple* stripe writes. To record and convey its mapping of data blocks to storage blocks (usually across multiple stripes), it uses an "association" that can include blocks from multiple stripes. One of ordinary skill in the art would understand the "association" to be a thing, given its noun form and use in the claims. For example, claim 21 says "as an association," clearly illustrating that it is a thing. Claim 39 recites, as a part of the storage system claimed, "an association generated in the file system," again showing it to be a thing. In the context of the '715 patent, one of ordinary skill in the art would understand that the "association" is a data structure whose purpose is to maintain and convey the assignment of certain data blocks to be stored in certain storage blocks, as plainly indicated in the

disputed claim terms. Claim 1, which includes the limitation "transmitting the association to a storage device manager…," further confirms this in indicating that the association is something that can be transmitted to another party such that it can be understood by the recipient. Given that usage and the association's contents (as described in the disputed claim terms), one of ordinary skill in the art would understand the association to be a "data structure".

97. Many other examples exist in the specification:

- The specification calls for both data blocks and an "association" to be transmitted, thus showing that the association is a data structure. *See*, *e.g.*, '715 patent at 13:14-23 ("The data blocks and the association are transmitted to, and processed by, the disk array manager 13 so that each data block is stored at its associated storage block in the group 120.")
- The specification explains that the "association" can take on "alternative embodiments," demonstrating that it is a data structure. *See id*_ at 13:34-43 ("In alternative embodiments of an association 15A, not all free storage blocks in the group of storage blocks are associated with buffered data blocks.").
- The specification uses the term "association" as a concrete noun and equates it to a "RAID map," which is understood in the art to be a data structure. *See id.* at 16:20-26. ("The association of a range or ranges of VBN's to objects at each level is sometimes referred to as a RAID map.").
- FIG. 2 depicts a disk array sending a two dimensional grid (*i.e.*, an "association") to the disk array manager. The rectangular grid being sent from the file system to the array has multiple discrete cells, thus depicting a data structure containing data in the separate cells of the grid. *See* '292 patent at Fig. 2; *see also id*. at Fig. 8 (depicting an "association" in even greater detail).

One of ordinary skill would understand that the two types of block (data block and storage block) discussed in the claims need not be the same size, and that the claim therefore does not require a one-to-one mapping between the data blocks and the one or more storage blocks. As discussed above, using different file system block sizes and stripe unit sizes is common. Indeed, the specification even notes that the file system could use a variable block size. '715 patent at 18:45-49. That a one-to-one mapping is not required is confirmed by the plain language of other claims. For example, claim 1 recites "associating *each data block* with *a respective one* of the storage blocks…" *See* '715 patent at 20:7-9. That the inventors claimed this type of one-to-one mapping in other claims is evidence that they knew how to do so. Claims that do not do so, such as the ones with the disputed terms, would not be interpreted as requiring such a one-to-one mapping by one of ordinary skill in the art.

98. Sun's opening brief argues that, "NetApp repeatedly argued to the Examiner that the claims of the '715 patent were distinguishable over the prior art because the claims require associating each data block with a respective one of the storage blocks. This construction [sic]… enabled NetApp to obtain the '715 patent…" *See* Sun Opening Cl. Constr. Br. at 35. It further argues that, "This construction [sic]… enabled NetApp to obtain the '715 patent." *Id.* Thus, Sun argues that the so-called "one-to-one" mapping of data blocks to storage blocks, which appears in its proposed construction, was a necessary point of novelty. The prosecution history, however, indicates that the opposite is true. NetApp's statements to the examiner indicate that NetApp viewed writes to a "plurality of stripes" and the "association" data structure as differentiators. The examiner's comments also indicate that the claims were accepted only after emphasis was placed on writes to a "plurality of stripes" and explicit associations. As discussed above, NetApp included the so-called "one-to-one" mapping terminology in some of its pending claims, but not others (*e.g.*, the claims with the disputed terms), indicating that it was for some and not others.

99. During the prosecution history, the examiner rejected certain claims on the grounds that the prior art DeKoning reference "teaches… associating each data block with a respective one of the storage blocks, for transmitting the association to a storage device manager for processing of the single write transaction." *See* Exh. U at 3. In response, the applicants distinguished DeKoning from "representative" application claim 17 as follows: "the DeKoning patent is legally precluded from anticipating the claimed invention under 35 U.S.C. § 102 because of the absence … of Applicant's 'associating each data block with a respective one of the storage blocks, for transmitting the association to a storage device manager for processing of the single write transaction.'" *See* Exh. V at 17; *see also id.* at 15 (Where NetApp argued that, "all art cited during prosecution… is completely silent regarding '***associating each data block with a respective one of the storage blocks***' as claimed." (emphasis in original)). More broadly, applicants argued that "DeKoning does not address associating data blocks with storage blocks, but instead merely discusses the use of buffering smaller write requests into a larger write request, such as a RAID stripe write." *Id.* at 18. NetApp further argued,

"Applicant goes one step further by mapping each data block of the single write request with a storage block across a ***plurality of stripes***...." *Id*. at 20. (emphasis in original). Thus, the point of distinction was not the supposed one-to-one relationship (which was relevant to original claim 17), but instead the broader "association" concept as well as the concept of selecting blocks, where possible, from across the plurality of stripes.

100. The examiner's subsequent explanation of reasons for allowance did not identify a single novel feature. Instead, it listed several features together. Thus, it does not indicate a specific point of novelty: "…the prior art does not further teach buffering write requests, associating each data block to be stored with a respective one of the storage blocks across the plurality of stripes for a single write operation, and transmitting this association to the storage device manager…" *See* Exh. W at 2.

V.

MATERIALS REVIEWED

101. The list of materials I reviewed is attached as Exh. C.

VI.

COMPENSATION

102. My compensation for consulting on this matter is $500 per hour plus expenses. My compensation does not depend on the substance of my opinions or the outcome of this dispute.

VII.

PREVIOUS TESTIMONY

103. My previous testimony experience is listed below:

Communique Laboratory, Inc. v. Citrix Systems, Inc. and Citrix Online, LLC

- Northern District of Ohio, Eastern Division 1:06-CV-0253
- Firm: Baker & Hostetler LLP
- Representing: _01 Communique Laboratory, Inc.
- Activities: Two depositions, claim construction tutorial, expert and rebuttal reports, and several declarations
- Disposition: (Currently stayed. No final disposition.)

I declare under penalty of perjury under the laws of the United States of America and the State of California that the foregoing is true and correct.

Dated: July 21, 2008

Gregory R. Ganger, Ph.D.